Báo cáo :

Hệ thống Server Management System

Sinh viên: Lê Anh Đức

**Mục lục**

[I. Các công nghệ sử dụng 2](#_Toc167425401)

[II. Mô tả hệ thống 2](#_Toc167425402)

[III. Cơ sở dữ liệu 3](#_Toc167425403)

[IV. Các chức năng 3](#_Toc167425404)

[a) Kiểm tra trạng thái định kỳ của server: 3](#_Toc167425405)

[b) Xác thực, phân quyền bằng JWT 4](#_Toc167425406)

[c) Tạo Server 5](#_Toc167425407)

[d) View server 7](#_Toc167425408)

[e) Update server 13](#_Toc167425409)

[f) Delete server 14](#_Toc167425410)

[i) Import servers 15](#_Toc167425411)

[j) Export servers 18](#_Toc167425412)

[h) Báo cáo 19](#_Toc167425413)

[V, Các yêu cầu phi chức năng 20](#_Toc167425414)

[1. Sử dụng OpenAPI 20](#_Toc167425415)

[2. Xác thực phân quyền bằng JWT 21](#_Toc167425416)

[3. Redis Cache 21](#_Toc167425417)

[4. Elasticsearch 22](#_Toc167425418)

[5. Log và Logrotate 22](#_Toc167425419)

[6. Sử dụng Kafka để invalid cache 23](#_Toc167425420)

[VI. Cách cài đặt: 24](#_Toc167425421)

Link github: https://github.com/minhmannh2001/vcs-sms

Đề bài: Công ty VCS hiện có khoảng 1000 server. Xây dựng 1 hệ thống quản lý trạng thái On/Off của danh sách server này.

# I. Các công nghệ sử dụng

* Ngôn ngữ: Golang, framework Gin
* Cơ sở dữ liệu: Postgresql, Elasticsearch
* Công cụ trực quan dữ liệu Kibana
* Message Queue: RabbitMQ
* Công nghệ đóng gói Docker
* Hệ thống cache Redis
* Công cụ xây dựng phát triển, thiết kế, xây dựng và làm tài liệu cho hệ thống OpenAPI - Swagger

# II. Mô tả hệ thống

A diagram of a computer network

Description automatically generated

Hệ thống gồm các thành phần chính:

- SMS Server: Máy chủ quản lý các máy chủ khác

- Server n: máy chủ thứ n trong hệ thống

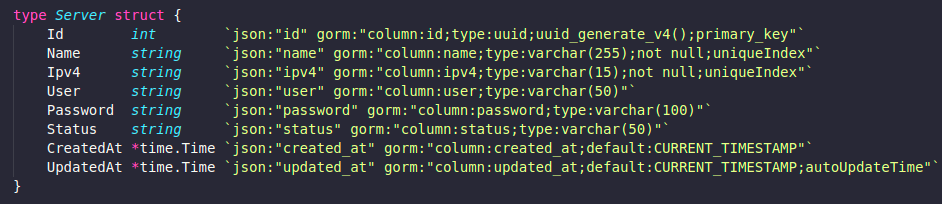
- Postgresql: Cơ sở dữ liệu

- Redis: Bộ đệm trong hệ thống

- Kafka: hàng đợi để invalid cache

- Elasticsearch: Lưu trữ các tín hiệu phản hồi từ các máy chủ khác, dùng để trích xuất thông tin liên quan đến trạng thái của các máy chủ trong báo cáo

# III. Cơ sở dữ liệu



Định nghĩa struct Server với các trường thông tin và các mô tả cụ thể dùng để migrate sang cơ sở dữ liệu thông qua thư viện gorm



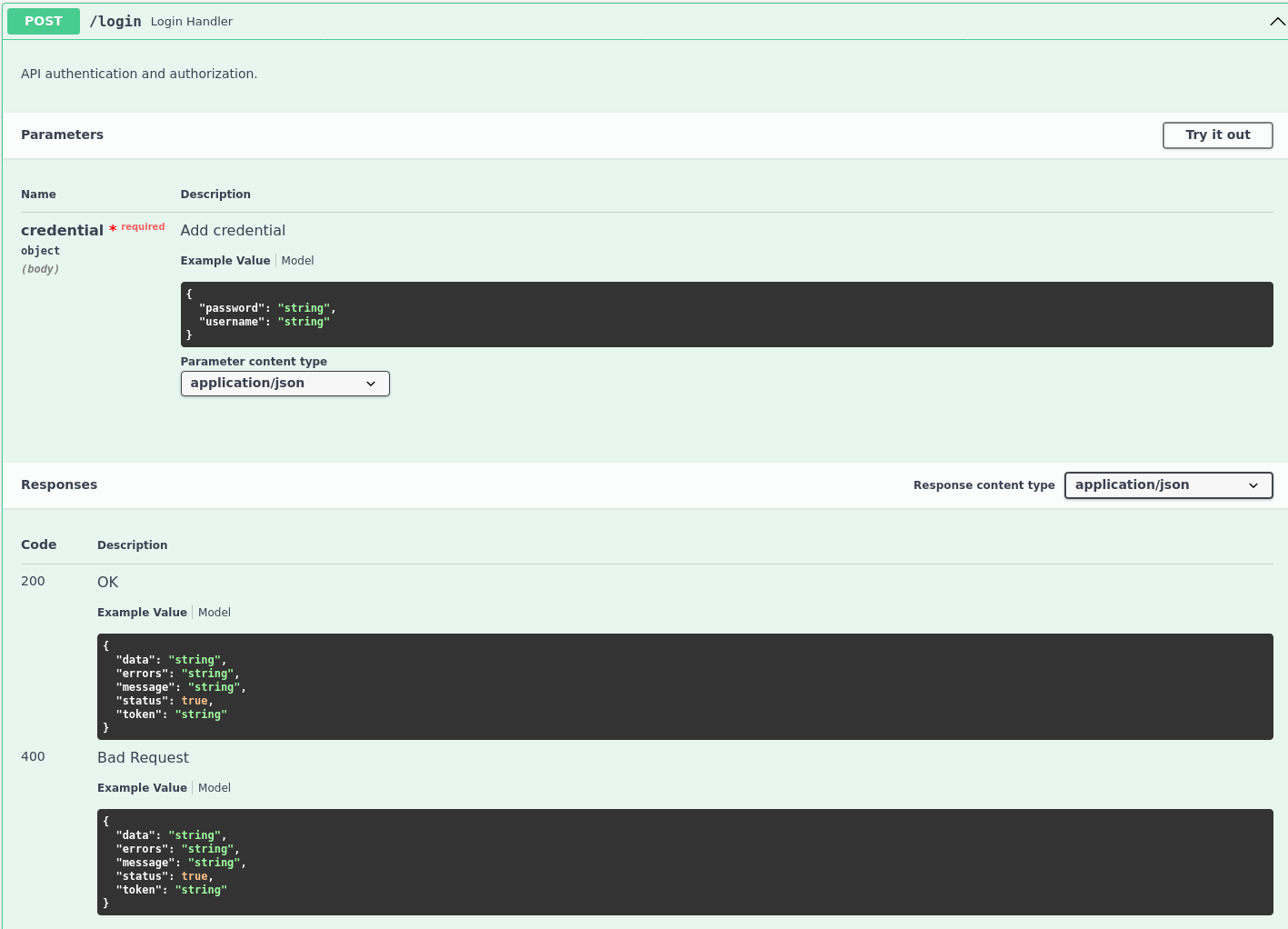
Bản mô tả dữ liệu trong CSDL Postgresql

# IV. Các chức năng

## a) Kiểm tra trạng thái định kỳ của server:

Cứ định kì 5p 1 lần, serivice healcheck sẽ gửi tín hiệu và check các server. Mỗi khi kiểm tra server thành công hoặc thất bại sẽ lưu log vào ES. Sau đó sẽ thực hiện truy vấn đến ES để thực hiện các yêu cầu phù hợp

## b) Xác thực, phân quyền bằng JWT



Hệ thống sử dụng cơ chế xác thực JWT, ta cần đăng nhập để lấy token trước khi thực hiện các chức năng khác trong hệ thống.

Demo

1. URL để xác thực

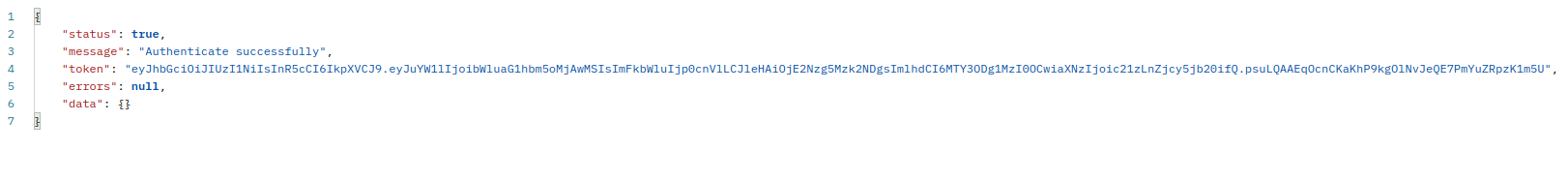


2. Body chứa json gồm username và password

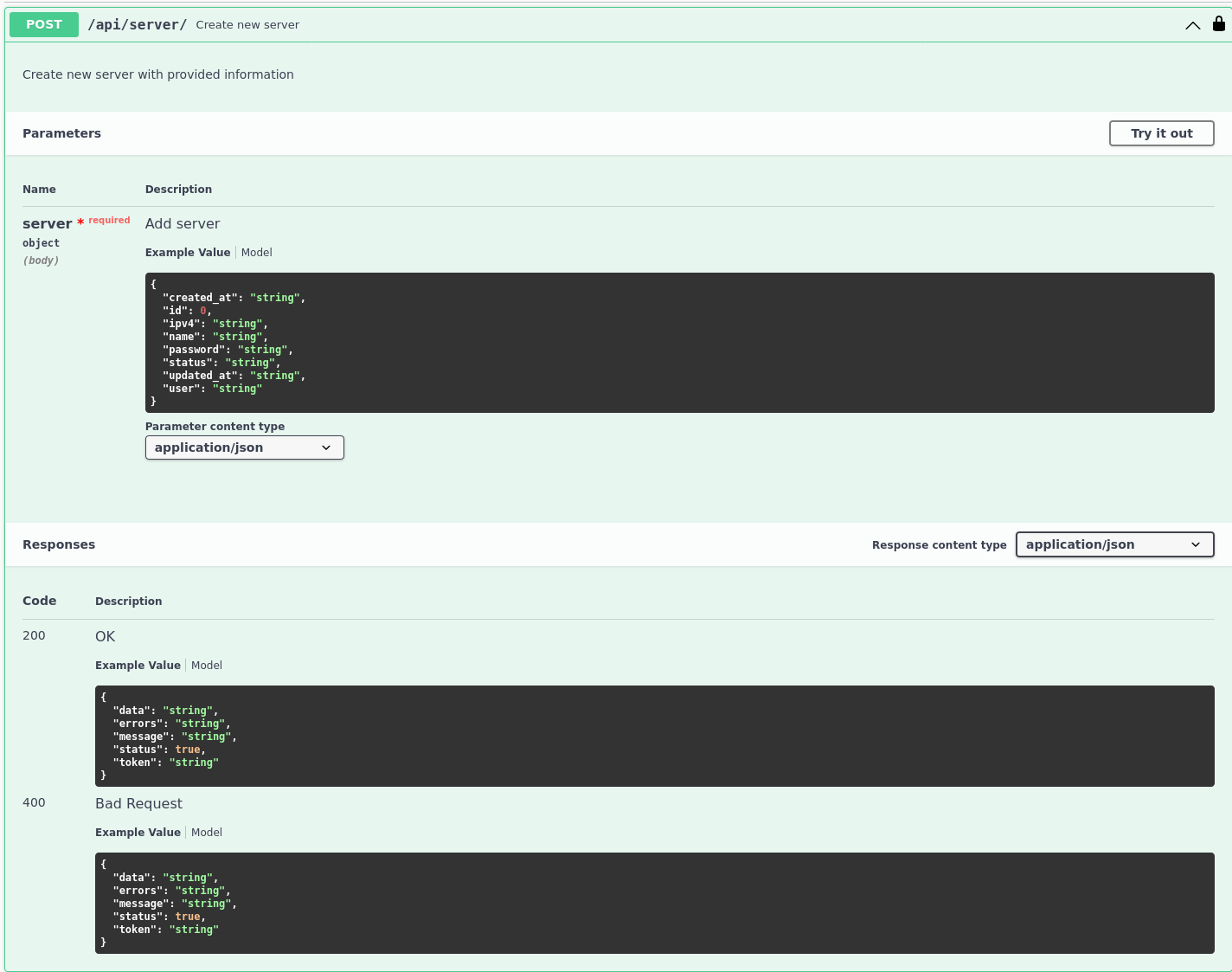
A screenshot of a computer screen

Description automatically generated

3. Sau khi gửi request, ta nhận được response chứa mã token



## c) Tạo Server



Mô tả: Cho phép người dùng tạo 1 server với đầy đủ thông tin

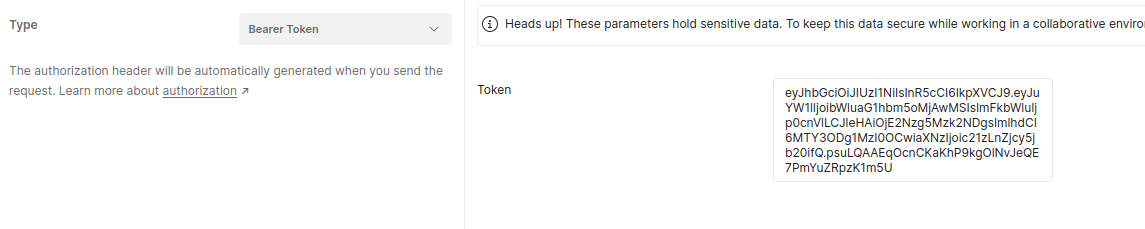
Input: thông tin server dạng json trong request body

Output: Kết quả tạo server

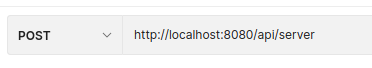
Bản mô tả API

Demo:

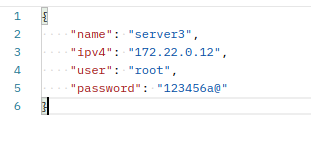
1. Thêm token JWT vào request



2. URL để gửi request

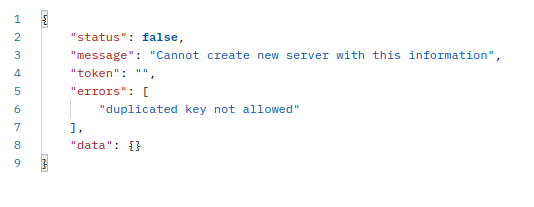


3. Nội dung phần body chứa thông tin về server sẽ thêm

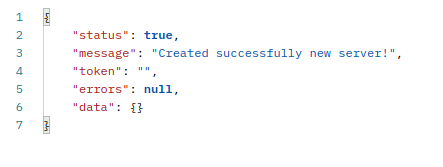


4. Nhận được response chứa message thông báo kết quả

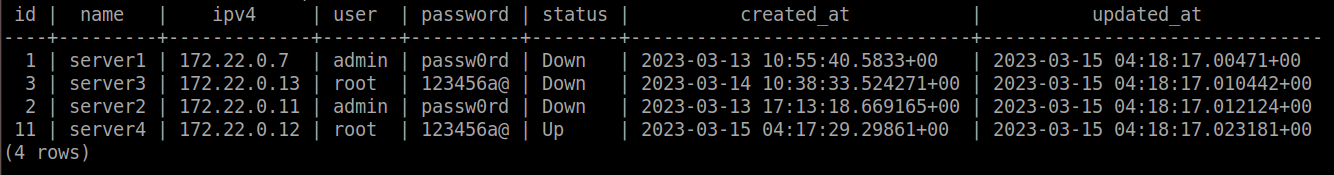
- Trường hợp lỗi: vì trong hệ thống đã có chứa server3 nên ta đổi tên thành server 4



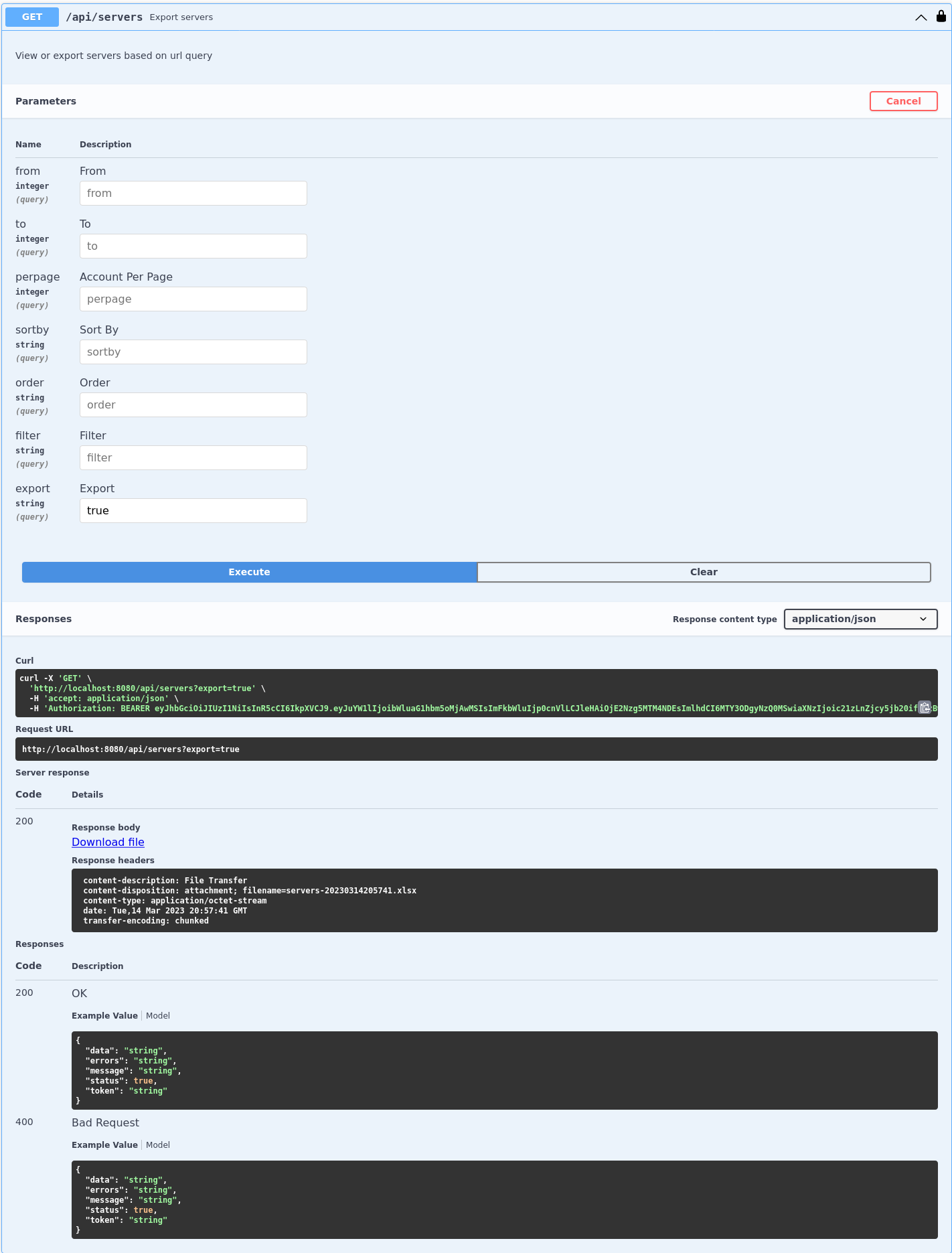
- Trường hợp thành công:



5. Kiểm tra trong cơ sở dữ liệu, ta thấy server 4 đã được thêm vào



## d) View server



Mô tả: Lấy ra danh sách server, có thể kèm filter nếu người dùng nhập vào. Danh sách được phân trang. Có sort theo trường nào đó.

Input:

* Filter (optional)
* From, to (optional): thông tin phân trang
* sort, order (optional): trường và thứ tự sort

Output: Danh sách server phù hợp

Demo:

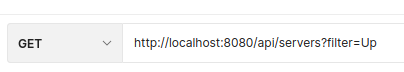
1. Thêm token JWT vào request

2. Phương thức và url để gửi request, có chứa các url params để điều chỉnh kết quả

- Nếu không chứa param, sẽ lấy ra tất cả server



- filter theo trạng thái up hoặc down





- filter theo trang, từ trang bao nhiêu đến trang bao nhiêu và số lượng kết quả mỗi trang



- sort theo trường và thứ tự sort



theo trường created\_at, thứ tự tăng dần



theo trường created\_at, thứ tự giảm dần

3. Kết quả trả về

- Trường hợp không có params, trả về tất cả



- Trường hợp filter theo trạng thái down



- Trường hợp filter theo trạng thái Up



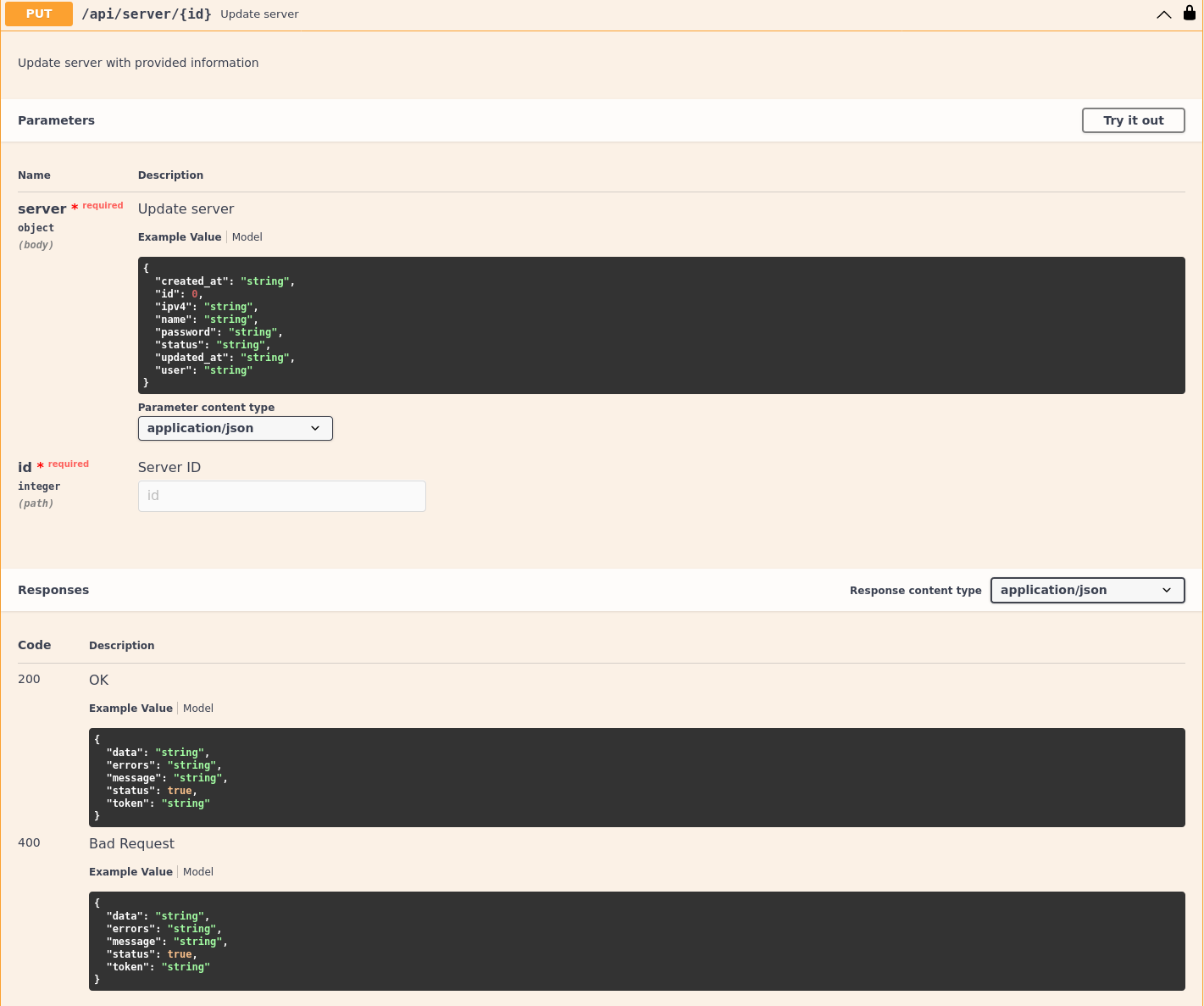
- filter từ trang 1 đến trang 2, mỗi trang 1 server



- filter theo trạng thái down, xếp theo trường created\_at theo chiều tăng dần



## e) Update server

Mô tả: cập nhật thông tin 1 server. Không cho phép cập nhật trường server\_id

Input:

* server\_id: server cần cập nhật
* update\_data: các thông tin cần update cho server

Output: Thông tin kết quả cập nhật

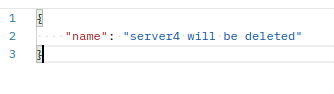
Demo:

1. Thêm token JWT vào request

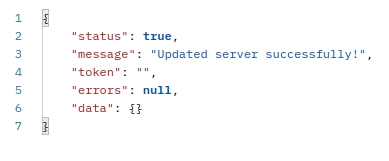
2. Phương thức và url để gửi request, thêm id của server cần update vào url



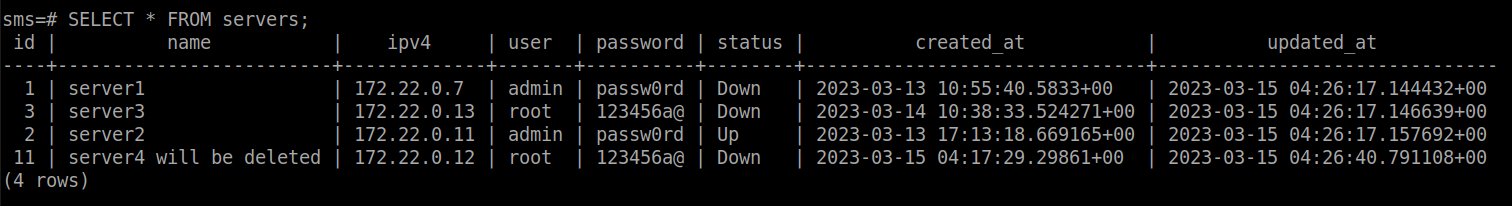
3. Nội dung body chứa thông tin cần chỉnh sửa



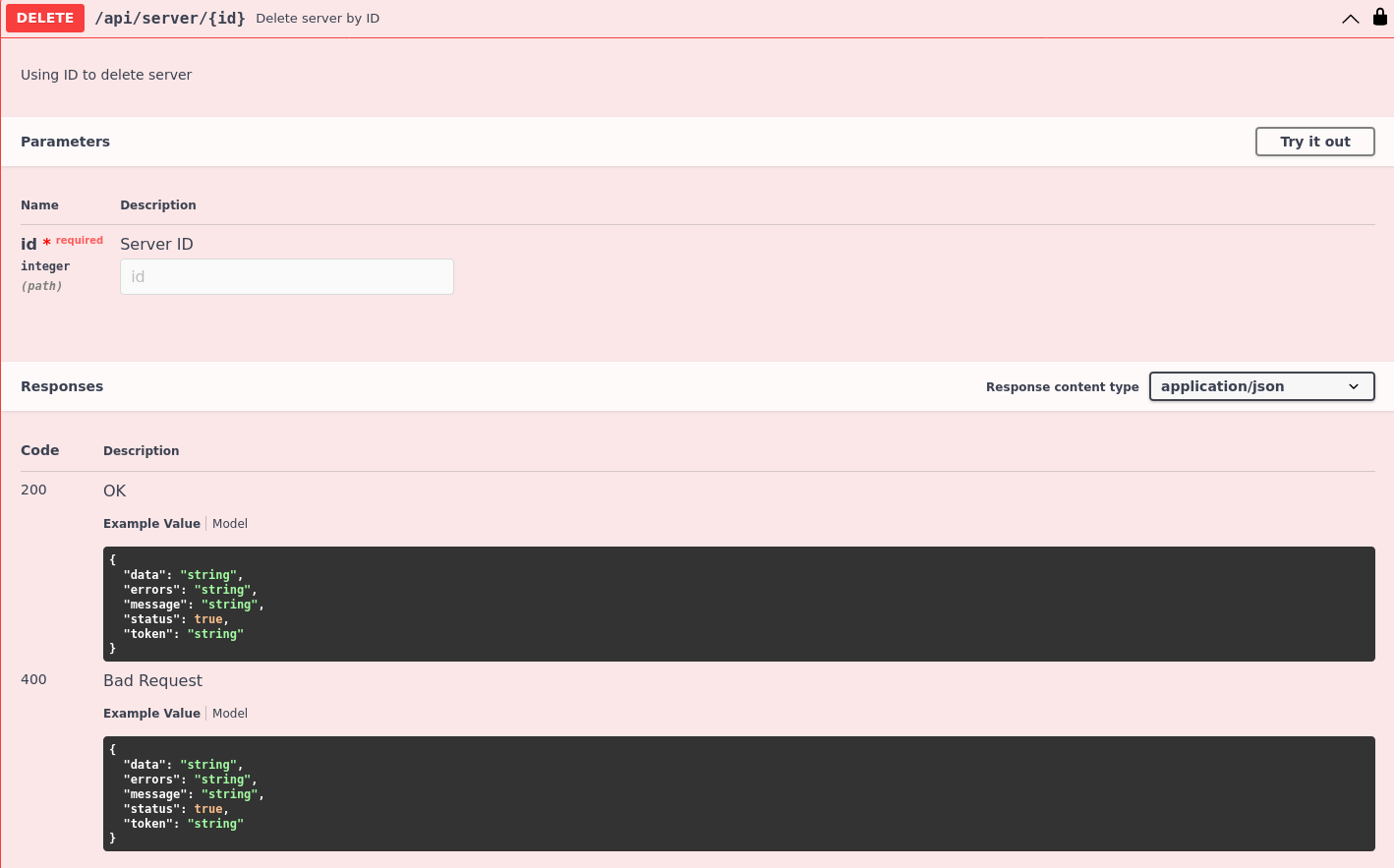
4. Kết quả request được trả về trong response



5. Kiểm tra lại trong CSDL, ta thấy tên của server đã thay đổi, ngoài ra trường updated\_at cũng thay đổi



## f) Delete server



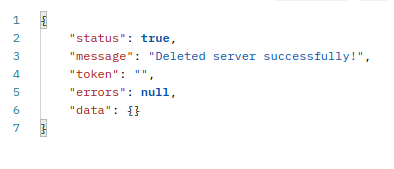
Demo:

1. Thêm token JWT vào request

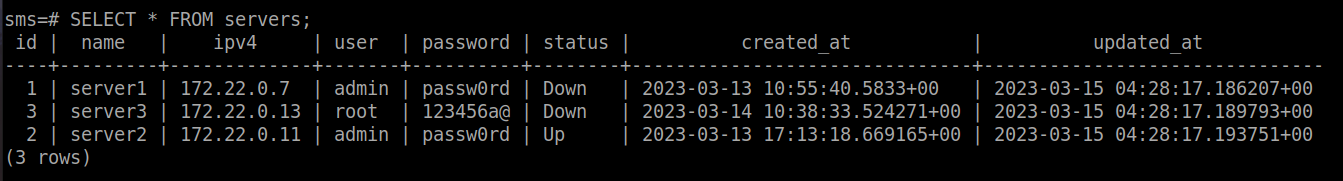
2. Phương thức và url để gửi request, ta thêm id của server cần xóa vào url



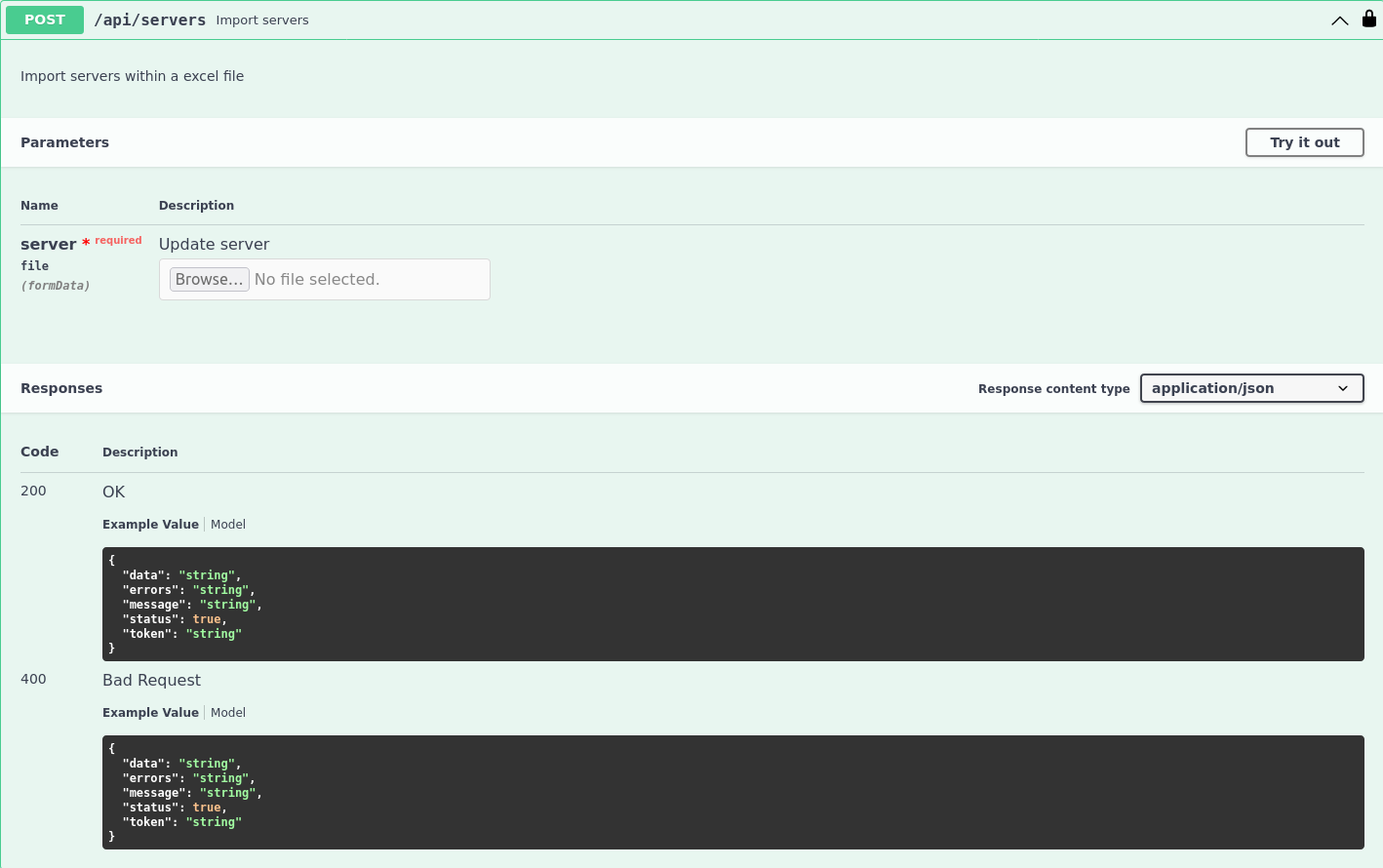
3. Kết quả trả về thông qua response



4. Kiểm tra lại CSDL, thấy server 4 đã bị xóa



## i) Import servers



Mô tả: cho phép tạo 1 danh sách nhiều server từ file excel. Bỏ qua các server\_id hoặc user\_name đã tồn tại

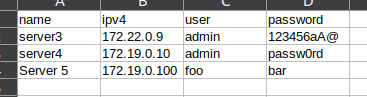
Input: File cần import

Output: Danh sách server đã import thành công, danh sách server import thất bại

Demo:

1. Thêm token JWT vào request

2. Chuẩn bị file để import



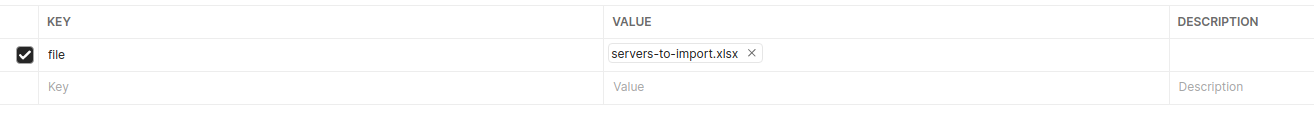
Nội dung:

* server3 bị trùng tên
* server 4 hợp lệ
* server 5 không thể kết nối tới, không tồn tại địa chỉ ip

3. Phương thức và url để gửi request



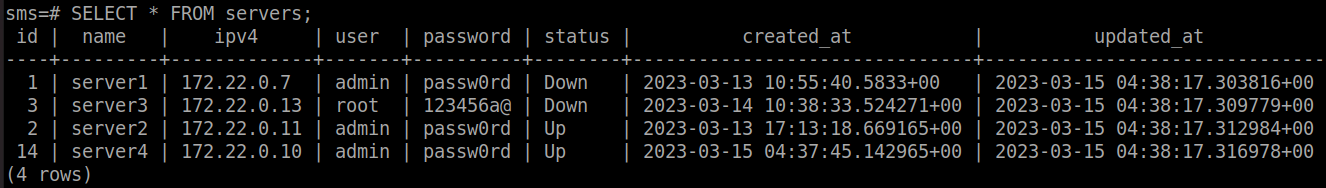
4. Chọn tệp để gửi lên, tệp nằm trong form-file của body



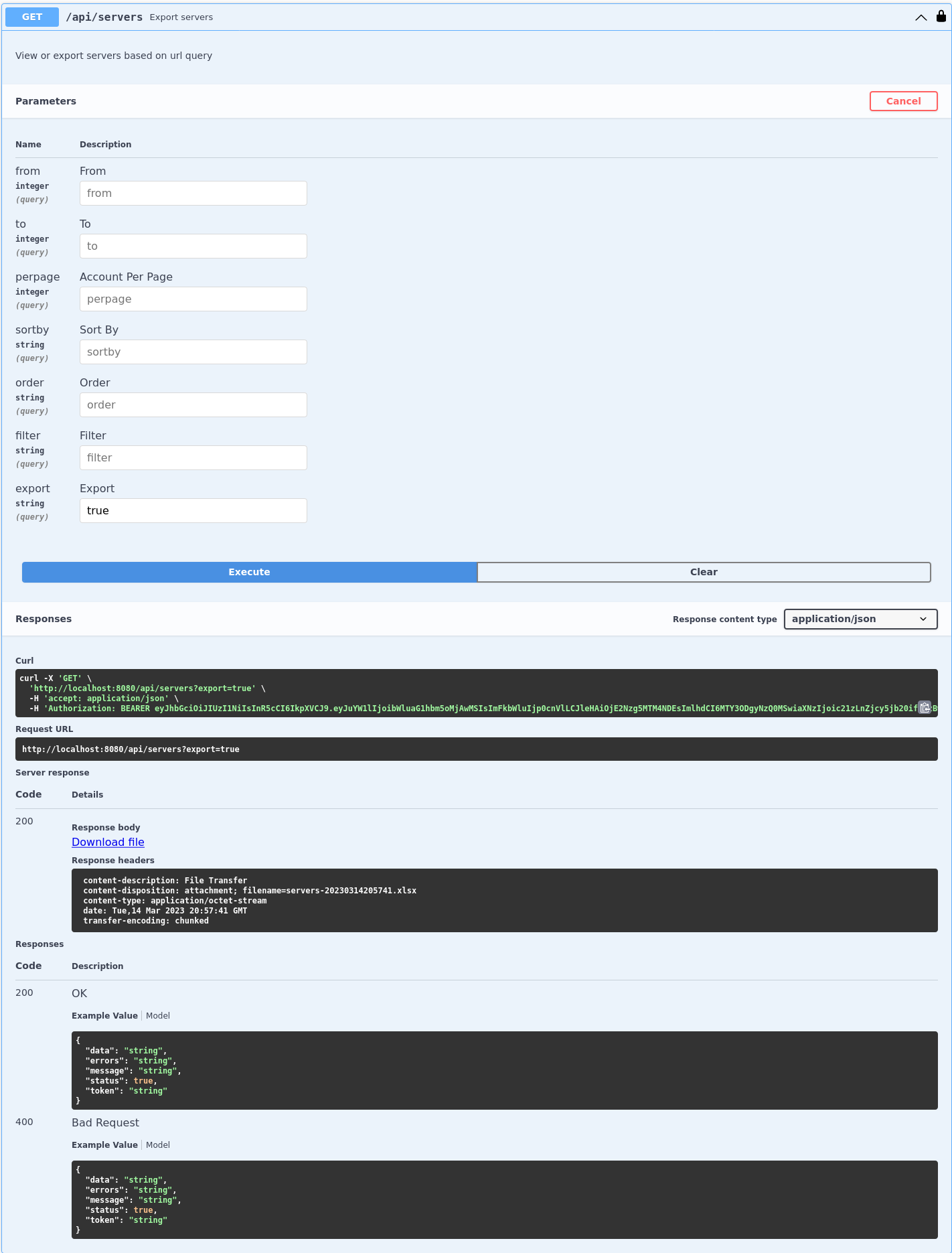
5. Kết quả gửi về qua response, bao gồm danh sách server thêm thành công và danh sách server không thể thêm vào CSDL



6. Kiểm tra lại CSDL, ta thấy server4 đã được thêm



## j) Export servers

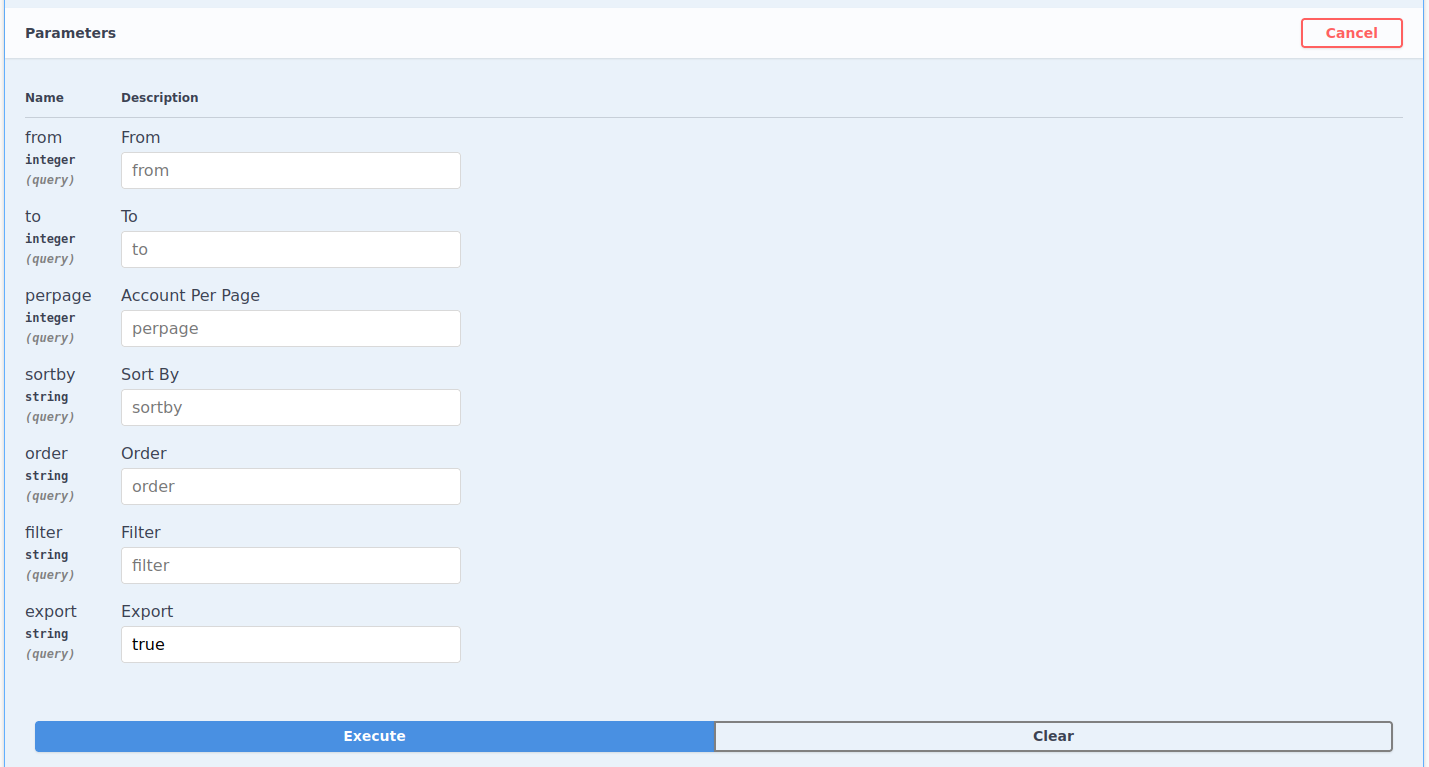


Mô tả: Cho phép export 1 danh sách ra file exel

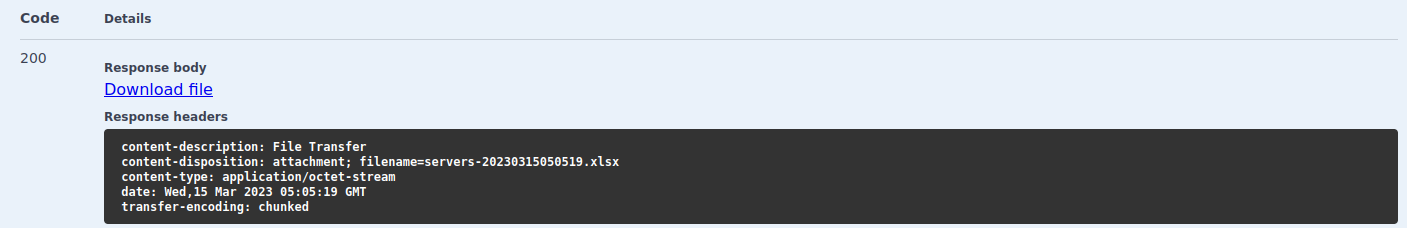
Demo:

1. Thêm token JWT vào request

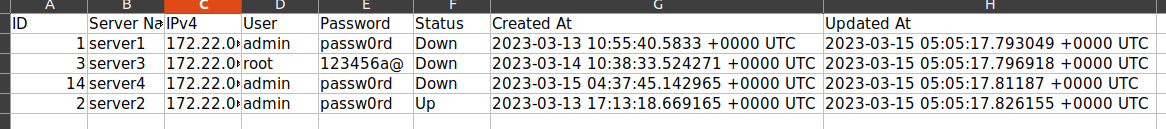
2. url và phương thức gửi request, chứa param ?export=true



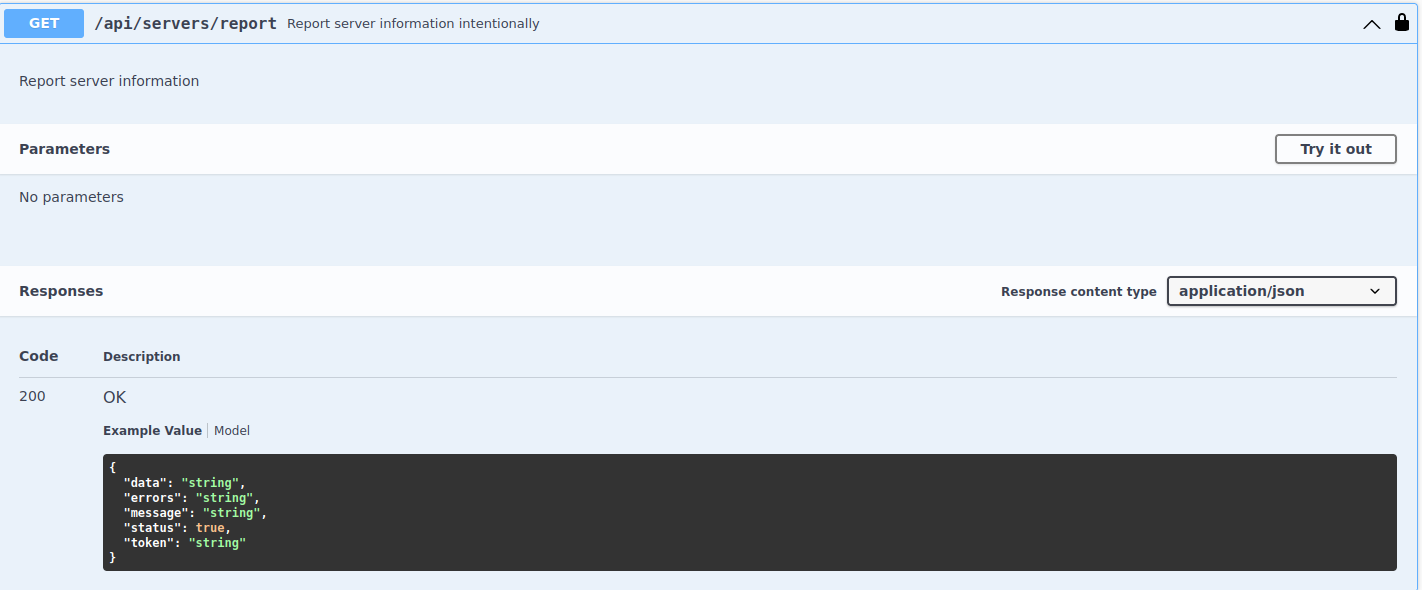
3. Kết quả trả về



4 Mở file ra kiểm tra, file chứa thông tin về các server



## h) Báo cáo



Định kỳ 30 phút thì gửi thông tin email cho người quản trị với thông báo trạng thái server trong vòng 3 tiếng vừa rồi:

* Các server trong hệ thống
* Số lượng server on
* Số lượng server off
* Tỉ lệ thời gian Uptime của server
* Các khoảng thời gian downtime của server

Xây dựng API để có thể chủ động report thông tin trên

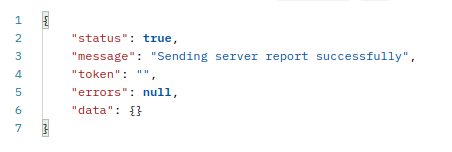
Demo:

1. Thêm token JWT vào request

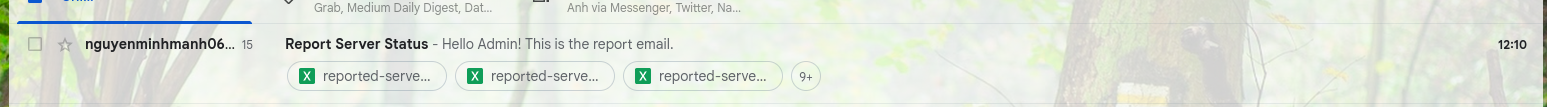
2. Phương thức và url để gửi request

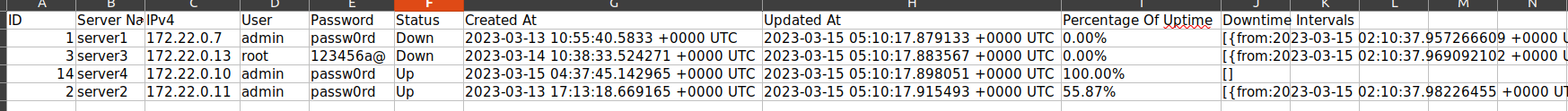


3. Kết quả trả về thông qua response



4. Kiểm tra email để xem thư báo cáo



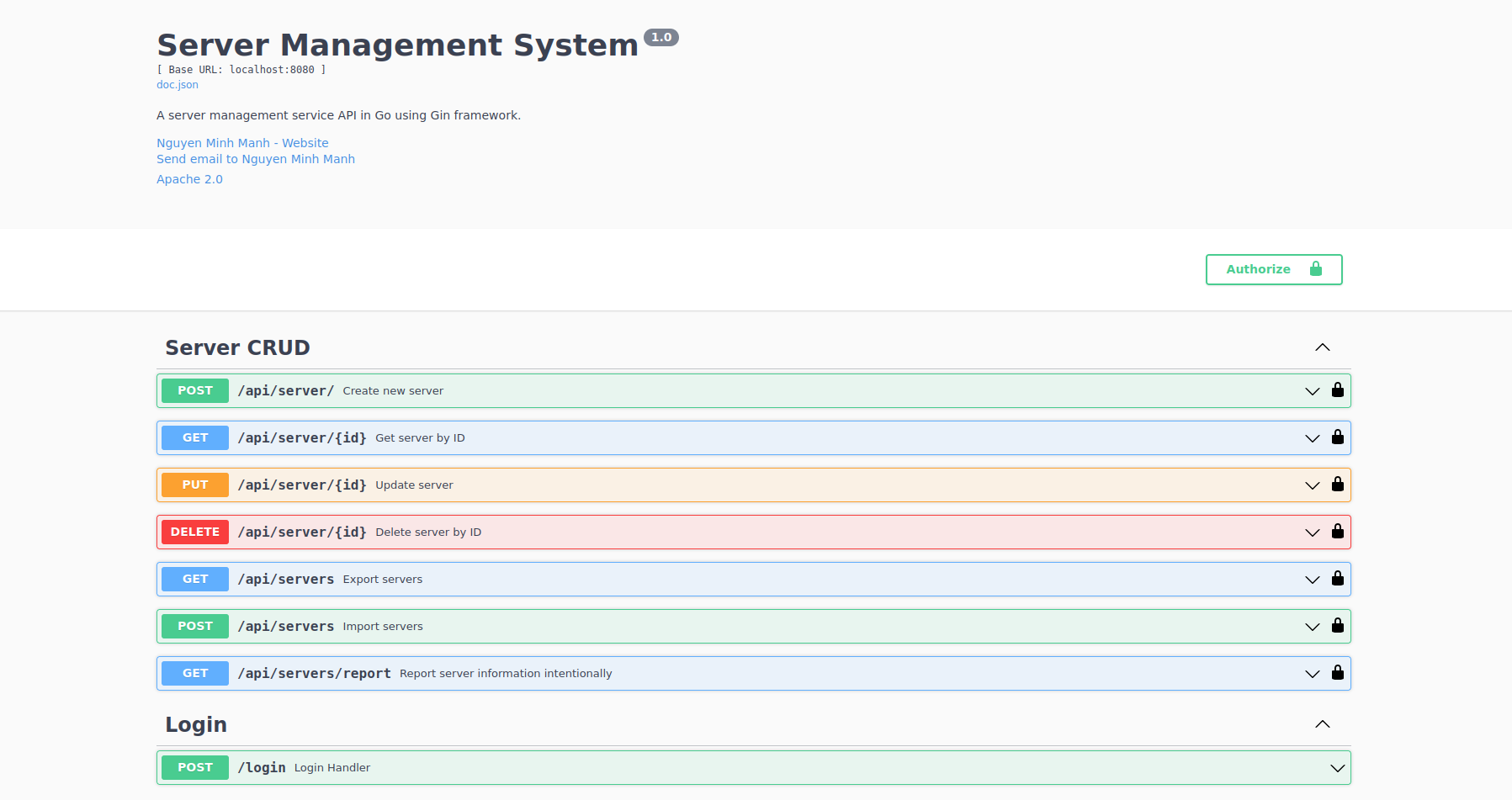


Ta thấy rằng server 4 từ lúc khởi tạo đến hiện tại chưa bị tắt nên trạng thái Uptime là 100%

# V, Các yêu cầu phi chức năng

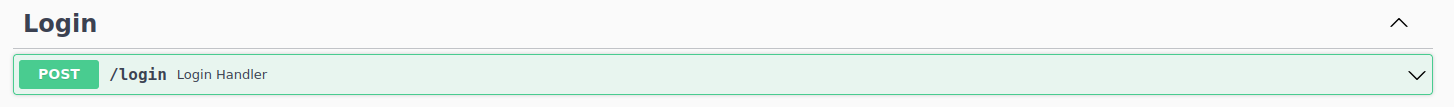
## 1. Sử dụng OpenAPI

Tuân thủ và sử dụng Swagger để tạo tài liệu OpenAPI cho từng endpoint



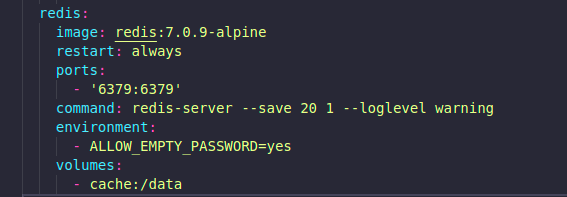
## 2. Xác thực phân quyền bằng JWT

Người dùng sử dụng endpoint /login để lấy token, cần thêm các token này vào các request để có thể thực hiện thành công

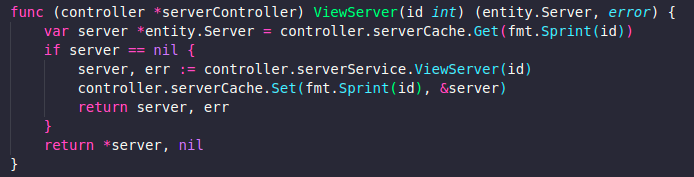


## 3. Redis Cache

Sử dụng Redis Cache để optimize performance cho các nghiệp vụ view server, export server



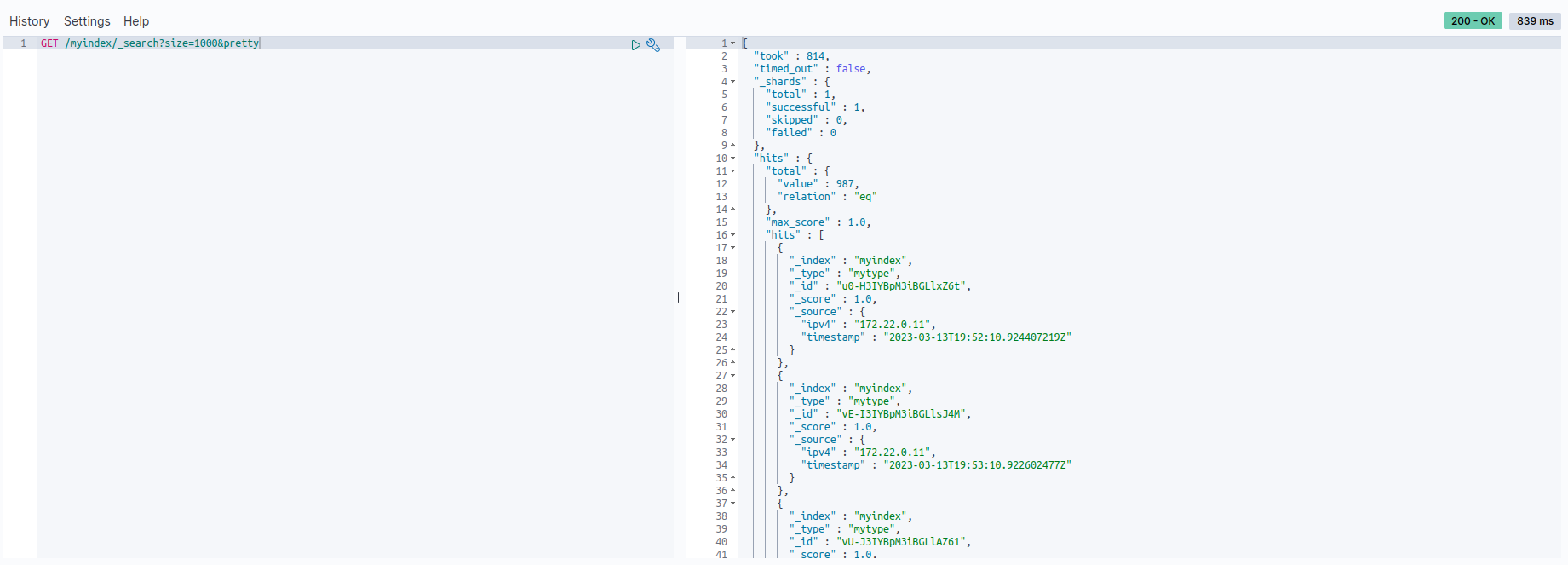
Thêm redis vào hệ thống



Thực hiện kiểm tra dữ liệu cache trước mỗi request

## 4. Elasticsearch

Dùng để lưu trữ các phản hồi từ các server đến SMS server, dùng để tổng hợp thông tin báo cáo định kỳ



## 5. Log và Logrotate

Sử dụng pakage lumberjack.Logger để tạo log rotate. Config logger

A screen shot of a computer

Description automatically generated

Khi file log đạt đủ 10MB thì file sẽ được backup và sau 28 ngày thì file backup sẽ bị xoá đi

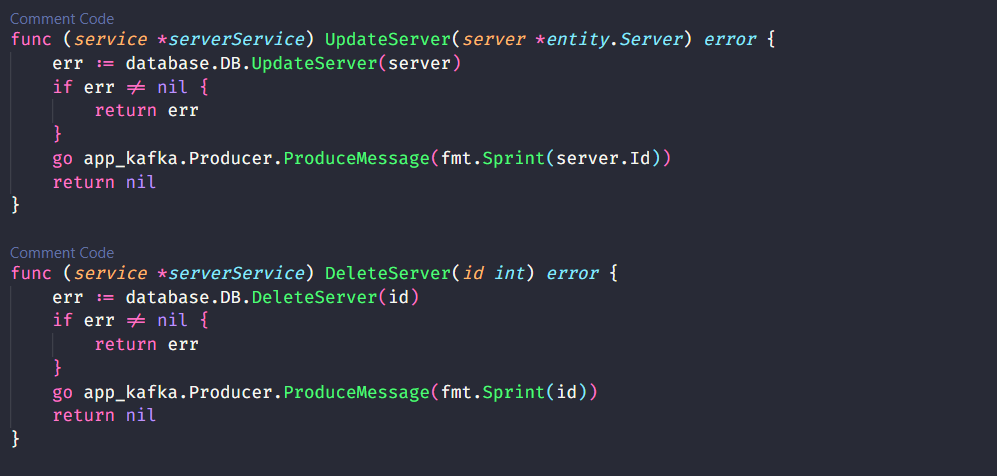
Nội dung log

A screenshot of a computer program

Description automatically generated

# 6. Sử dụng Kafka để invalid cache

Sử dụng kafka để invalid cache, mỗi khi database có thay đổi về server(update, delete) thì sẽ gửi 1 message vào kafka.



Sau đó 1 service Kafka chạy ngầm sẽ lắng nghe và xử lý message để invalid cache

A computer screen shot of a program code

Description automatically generated